목차

Last update: 2025/12/24 01:19	폰트이름으로파일명변경	https://atl.kr/dokuwiki/doku.php/%ED%8F%B0%ED%8A%B8_%EC%9D%B4%EB%A6%84%EC%9C%BC%EB%A1%9C_%ED%8C%8C%EC%9D%BC%EB%AA%85_%EB%B3%80%EA%B2%BD

폰트 이름으로 파일명 변경

— *이강우 2025/12/24 01:17*

폰트파일의 정보를 읽어서 해당 정보를 기반으로 파일명을 변경해주는 스크립트

아래 스크립트가 동작하기 위해서는 `fc-scan` 명령어가 필요하다. `fc-scan`명령어는 `fontconfig` 패키지에 포함되어있다.(RHEL/Rocky 기준)

```
#!/bin/bash

# 로케일 설정 (한글 및 유니코드 바이트 계산 필수)
export LC_ALL=C.UTF-8
export LANG=C.UTF-8

# 설정: 폰트 디렉토리 (필요시 수정)
FONTDIR="./fonts/"

# 색상 출력 정의
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
PURPLE='\033[0;35m'
BOLD='\033[1m'
NC='\033[0m'

# === 도움말 출력 함수 ===
function show_help() {
    echo -e
"${BLUE}=========================================================${NC}"
    echo -e "${GREEN}${BOLD}          폰트 파일 이름 자동 정리 스크립트${NC}"
    echo -e
"${BLUE}=========================================================${NC}"
    echo -e "폰트의 메타데이터(Fullname)를 분석하여 파일명을 정리합니다."
    echo ""
    echo -e "${YELLOW}사용법:${NC}"
    echo -e "  $0 ${BOLD}[verify | rename]${NC}"
    echo ""
    echo -e "${YELLOW}명령어 상세:${NC}"
    echo -e "  ${BOLD}verify${NC}  : ${GREEN}검증 모드${NC} (변경될 내용을 미리 확
인합니다.)"
    echo -e "  ${BOLD}rename${NC}  : ${RED}실행 모드${NC} (실제로 파일 이름을 변경
합니다.)"
    echo ""
    echo -e "${YELLOW}도움말:${NC}"
    echo -e "  ${BOLD}-h, --help${NC} : 이 안내 문구를 출력합니다."
    echo ""
    echo -e "${YELLOW}비고:${NC}"
    echo -e "  - 인자값 없이 실행하면 이 도움말이 출력됩니다."
```

```
    echo -e "  - 한글 이름을 우선적으로 선택합니다."
    echo -e "  - 파일명이 32바이트를 초과하면 첫 단어만 추출하고 'VF'를 붙입니다."
    echo -e "  - 이미 'VF'가 붙어있는 경우 중복해서 붙이지 않습니다."
    echo -e
"${BLUE}=====================================================${NC}"
}

# === 인자값 처리 ===
if [[ -z "$1" ]]; then
    show_help
    exit 0
fi

case "$1" in
    "verify")
        MODE="dry-run"
        echo -e "${YELLOW}====================================${NC}"
        echo -e "${GREEN}[검증 모드] 미리보기만 수행합니다 (파일 변경 없음)${NC}"
        echo -e "${YELLOW}====================================${NC}"
        ;;
    "rename")
        MODE="real"
        echo -e "${YELLOW}====================================${NC}"
        echo -e "${RED}[실제 변경 모드] 파일명을 수정합니다!${NC}"
        echo -e "${YELLOW}====================================${NC}"
        ;;
    "-h"|"--help")
        show_help
        exit 0
        ;;
    *)
        echo -e "${RED}[!] 알 수 없는 인자입니다: $1${NC}"
        show_help
        exit 1
        ;;
esac

# 폰트 파일 검색
font_files=$(find "${FONTDIR}" -type f \( -name "*.ttf" -o -name "*.otf" -o
-name "*.ttc" -o -name "*.woff" -o -name "*.woff2" \) 2>/dev/null)

if [ -z "$font_files" ]; then
    echo -e "${RED}[!] '${FONTDIR}'에서 폰트 파일을 찾을 수 없습니다.${NC}"
    exit 0
fi

processed=0
```

```
renamed=0

IFS=$'\n'
for original_filepath in $font_files; do
    ((processed++))
    echo "--------------------------------------"
    echo -e "${YELLOW}[*] 파일 확인: $original_filepath${NC}"

    # 1. fc-scan 정보 추출
    fullname_string=$(fc-scan -f "%{fullname}" "$original_filepath"
2>/dev/null)
    echo -e "  ${PURPLE}[DEBUG] fc-scan raw value:${NC} '$fullname_string'"

    if [ -z "$fullname_string" ]; then
        echo -e "  ${RED}[!] fullname 정보를 가져올 수 없습니다. 건너뜀.${NC}"
        continue
    fi

  # 2. 토큰 분리 및 한글 우선 선택
    IFS=',' read -ra tokens <<< "$fullname_string"
    selected_name=""
    first_token=""

    for i in "${!tokens[@]}"; do
        token="${tokens[$i]}"
        token=$(echo "$token" | sed -e 's/^[[:space:]]*//' -e
's/[[:space:]]*$//')
        [ -z "$token" ] && continue
        [ -z "$first_token" ] && first_token="$token"

        if echo "$token" | perl -CS -ne 'exit 1 unless /[\x{AC00}-
\x{D7A3}\x{3131}-\x{318E}]/;' ; then
            [ -z "$selected_name" ] && selected_name="$token"
            echo -e "  ${BLUE}[DEBUG] Token[$i]:${NC} '$token' ${GREEN}(한글
포함)${NC}"
        else
            echo -e "  ${BLUE}[DEBUG] Token[$i]:${NC} '$token' (영문/기타)"
        fi
    done

  # 3. 이름 정제
    new_name_base="${selected_name:-$first_token}"
    sanitized=$(echo "$new_name_base" | sed 's/[[:space:]]/_/g' | sed
's/[\\/:\*\?"<>|]//g' | sed 's/___*/_/g')

  # 4. 32바이트 초과 처리 및 VF 중복 방지
    byte_size=$(printf "%s" "$sanitized" | wc -c)
    if [ "$byte_size" -gt 32 ]; then
        first_word="${new_name_base%% *}"
        sanitized_short=$(echo "$first_word" | sed 's/[[:space:]]/_/g' | sed
's/[\\/:\*\?"<>|]//g')
```

```
        if [[ "$sanitized_short" =~ [vV][fF]$ ]]; then
            sanitized="$sanitized_short"
            echo -e "  ${YELLOW}[!] 바이트 초과 및 이미 VF 포함됨 -> 유지:${NC}
'$sanitized'"
        else
            sanitized="${sanitized_short}VF"
            echo -e "  ${YELLOW}[!] 바이트 초과 -> VF 추가:${NC} '$sanitized'"
        fi
    fi

    if [ -z "$sanitized" ]; then continue; fi

   # 5. 경로 및 중복 확인
    dir_path=$(dirname "$original_filepath")
    extension="${original_filepath##*.}"
    final_new_filepath="${dir_path}/${sanitized}.${extension}"

    counter=1
    while [ -e "$final_new_filepath" ] && [ "$final_new_filepath" !=
"$original_filepath" ]; do
final_new_filepath="${dir_path}/${sanitized}(${counter}).${extension}"
        ((counter++))
    done

   # 6. 실행
    if [ "$original_filepath" = "$final_new_filepath" ]; then
        echo "  [*] 이름 변경 불필요 (기존과 동일)"
    else
        echo -e "  ${GREEN}[+] 변경 대상:${NC}"
        echo "      FROM: $(basename "$original_filepath")"
        echo "      TO:   $(basename "$final_new_filepath")"
        if [[ "$MODE" == "real" ]]; then
            mv "$original_filepath" "$final_new_filepath"
            echo -e "  ${GREEN}[✓] 실제 변경 완료${NC}"
        fi
        ((renamed++))
    fi
done

echo "--------------------------------------"
echo -e "처리 완료! (총 $processed 건 중 $renamed 건 변경 대상)"

if [[ "$MODE" == "dry-run" && $renamed -gt 0 ]]; then
    echo ""
    echo -e "${YELLOW}확인 후 실제 적용하시려면 다음 명령어를 입력하세요:${NC}"
    echo -e "${GREEN}${BOLD}$0 rename${NC}"
fi
```